M.S. THESIS

Safe Reinforcement Learning for Probabilistic Safety Verification: A Lyapunov-Based Approach

확률적 안전성 검증을 위한 안전 강화학습: 랴푸노브 기반 방법론

BY

Subin Huh

AUGUST 2020

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

M.S. THESIS

Safe Reinforcement Learning for Probabilistic Safety Verification: A Lyapunov-Based Approach

확률적 안전성 검증을 위한 안전 강화학습: 랴푸노브 기반 방법론

BY

Subin Huh

AUGUST 2020

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Safe Reinforcement Learning for Probabilistic Safety Verification: A Lyapunov-Based Approach

확률적 안전성 검증을 위한 안전 강화학습: 랴푸노브 기반 방법론

지도교수 양 인 순

이 논문을 공학석사 학위논문으로 제출함

2020 년 8 월

서울대학교 대학원

전기·정보공학부

Subin Huh

Subin Huh의 공학석사 학위논문을 인준함

2020 년 8 월

위 원 장 _____

부위원장 _____

위 원 _____

Abstract

Emerging applications in robotic and autonomous systems, such as autonomous driving and robotic surgery, often involve critical safety constraints that must be satisfied even when information about system models is limited. In this regard, we propose a model-free safety specification method that learns the maximal probability of safe operation by carefully combining probabilistic reachability analysis and safe reinforcement learning (RL). Our approach constructs a Lyapunov function with respect to a safe policy to restrain each policy improvement stage. As a result, it yields a sequence of safe policies that determine the range of safe operation, called the *safe set*, which monotonically expands and gradually converges. We also develop an efficient safe exploration scheme that accelerates the process of identifying the safety of unexamined states. Exploiting the Lyapunov shieding, our method regulates the exploratory policy to avoid dangerous states with high confidence. To handle high-dimensional systems, we further extend our approach to deep RL by introducing a Lagrangian relaxation technique to establish a tractable actor-critic algorithm. The empirical performance of our method is demonstrated through continuous control benchmark problems, such as a reaching task on a planar robot arm.

Keywords: Safe reinforcement learning, probabilistic reachability analysis, safety specification

Student Number: 2018-25960

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Reachability and safety specifications for robotic and autonomous systems are one of fundamental problems for the verification of such systems. It is difficult to imagine deploying robots, without (safety) verification, in practical environments due to possible critical issues such as collisions and malfunctions. Several reachability analysis techniques have been developed for the safe operation of various types of systems (e.g., [1–3]) and applied to quadrotor control [4], legged locomotion [5], obstacle avoidance [6], among others. However, the practicality of these tools is often limited because they require knowledge of system models. The focus of this work is to develop a model-free reinforcement learning method for specifying reachability and safety in a probabilistic manner.

Several learning-based safety specification methods have recently been proposed for *deterministic* dynamical systems without needing complete information about system models. To learn backward reachable sets, Hamilton–Jacobi reachability-based tools were used in conjunction with Gaussian process regression [7] and reinforcement learning [8]. As another safety certificate, a region

of attraction was estimated using Lyapunov-based reinforcement learning [9] and a neural network Lyapunov function [10]. Forward invariance has also been exploited for safety verification by learning control barrier functions [11, 12].

Departing from these tools for deterministic systems, we propose a model-free safety specification method for stochastic systems by carefully combining probabilistic reachability analysis and reinforcement learning. Specifically, our method aims to learn the maximal probability of avoiding the set of unsafe states. Several methods have been developed for computing the probability of safety in various cases via dynamic programming when the system model is known [1, 13–15]. To overcome this limitation, our tool uses model-free reinforcement learning for estimating the probability of safety. We further consider safety guarantees during the learning process so that our scheme runs without frequent intervention of a human supervisor who takes care of safety. To attain this property, we employ the Lyapunov-based RL framework proposed in [16], where the Lyapunov function takes the form of value functions, and thus safety is preserved in a probabilistic manner through the Bellman recursion. We revise this safe RL method to enhance its exploration capability. Note that the purpose of exploration in our method is to enlarge or confirm knowledge about safety, while most safe RL schemes encourage exploration to find reward-maximizing policies within verified safe regions [17–19].

The main contributions of this work can be summarized as follows. First, we propose a safe RL method that specifies the probabilistic safety of a given Markov control system without prior information about the system dynamics. Our approach yields a sequence of safe and improving policies by imposing the Lyapunov constraint in its policy improvement stage and establishing a Lyapunov function in the policy evaluation stage. If there is no approximation error, our RL-based safety specification algorithm is guaranteed to run safely

throughout the learning process. In such a case, the safe region determined by our approach also monotonically expands in a stable manner, and eventually converges to the maximal safe set. Second, we develop an efficient safe exploration scheme to learn safe or reachable sets in a sample-efficient manner. Safe policies tend to avoid reaching the borders of safe regions, so the "learned" probability of safety at their borders and outside them is likely to be more inaccurate than others. To mitigate the imbalance of knowledge, we select the least-safe policy to encourage exploration. This exploratory policy visits less-safe states so that the safe set becomes more accurate or grows faster. Third, we implement our approach with deep neural networks to alleviate the scalability issue that arises in high-dimensional systems. Converting the Lyapunov constraints to a regularization term, our approach can be implemented in conventional actor-critic algorithms for deep RL. We further show that our method outperforms other baseline methods through simulation studies.

In addition, we note that reinforcement learning is an effective tool for the stochasticity and uncertainties in the system dynamics, but is not suitable for detecting the unsafe states in reality. Since a single visit to the undesirable states causes a great loss, the unsafe states should be identified before the learning process. Therefore, our scheme considers an experimental situation where the reachable hazards are small or even virtual. Still, safe learning property is useful as it helps to avoid excessive failures, which often lead to the waste of resources in every circumstance.

Chapter 2

Related work

The proposed framework aims to achieve two goals: (a) probabilistic reachability analysis in a model-free manner, (b) safe learning during the analysis. On one hand, there exists a group of safe RL algorithms that target goals similar to (a), *i.e.* approximating the measure of safety via fitting a generic framework like deep neural network [8, 10]. However, safe learning is not included in their interests, so the aforementioned algorithms cannot be fairly paralleled with ours. If the chances of drawing state transitions from a simulator are equally provided, they are expected to be more sample-efficient than ours since they are free from the burden of safe learning.

On the other hand, one can adapt the several approaches dealing with (b), which are developed to solve control problems with safety-related constraints, to perform safety analysis. To be specific, the assorted concepts of (un)safety, ranging from asymptotic stability [20], forward invariance [21] to the region of attraction [17, 19], are not compatible with our setting. The concept of probabilistic safety has been occasionally adopted in the literature [22], but it can be

interpreted as a specific kind of expected return [13], and thus the schemes based on constrained MDP (CMDP) framework [23–25] can be applied for the sake of probabilistic reachability analysis. Both objective and constraint terms are represented as cumulative sums of costs and thus can be converted to a probability of safety as covered in Chapter 3. Nevertheless, most of the CMDP-based safe RL algorithms are not suitable for meeting the standard of safe learning. As stated in Chapter 3.2, according to our definition of safe learning, a bundle of state-wise constraints should be imposed explicitly on the process of policy improvement. The number of constraints are equal to the size of the current safe set, whereas existing algorithms are often limited to the fixed number of constraints.

The fundamental feature of our approach is that we define a Lyapunov function as a *value function* of the current policy using the cost-shaping technique of [16]. Therefore, this is suited for an unknown stochastic system where the form of Lyapunov function cannot be determined in the first place. If one aims to achieve safe learning in a specific control system, the Lyapunov function or control barrier function can be chosen as [11, 12]. In general, the value function is a universal candidate of the Lyapunov function, which can be computed directly in model-based approaches [9]. Our choice of Lyapunov function design adds cost-shaping to the idea of using value function, thus it can be considered as a generalization. Also, reinforcement learning is an substitute for (approximate) dynamic programming in *model-free* settings. Of course the Lyapunov function can be learned from scratch as in [10]; however, it assumes that the state-wise safety is accessible for all the states, disregarding constraints in observation, unlike the reinforcement learning framework. On the other hand, we consider the case where an environment only provides local information through the agent. In this regard, our approach is more suitable for realistic situations.

Chapter 3

Background

We consider an MDP, defined as a tuple $(\mathcal{S}, \mathcal{A}, p)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function. We also use the notation $\mathcal{S}_{\text{term}}$ and $\mathcal{S}'$ to represent the set of termination states and non-terminal states, respectively. Moreover, a (stochastic) Markov policy, $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, is a measurable function, and $\pi(\boldsymbol{a}|\boldsymbol{s})$ represents the probability of executing action $\boldsymbol{a}$ given state $\boldsymbol{s}$. We also let Π denote the set of stochastic Markov policies.

3.1 Probabilistic Reachability and Safety Specifications

To define our problem formally, we first consider an episode is safe if an MDP does *not* visit a pre-specified *target set* $\mathcal{G} \subseteq S$ before arriving at a terminal state. The probability of such an event is denoted by *probability of safety* throughout this article. Target set $\mathcal{G}$ can be interpreted as the set of forbidden states, representing the undesirable conditions in a system. Conversely, $\mathcal{G}^c$ is the set of

available states, although a state in this set might transit into $\mathcal{G}$ depending on either a policy or an MDP. We use the complement to compute the probability of safety. That is, we deal with the problem of evaluating the probability of visiting the target set at least once given an initial state $\boldsymbol{s}$ and a Markov policy π:

$$P_{\boldsymbol{s}}^{\text{reach}}(\pi) := \mathbb{P}^{\pi}\left(\exists t \in \{0, \ldots, T^* - 1\} \text{ s.t. } s_t \in \mathcal{G} | s_0 = \boldsymbol{s}\right),$$

where T^* is the first time to arrive at a terminal state. Note that $P_{\boldsymbol{s}}^{\text{reach}}(\pi)$ represents the *probability of unsafety*. Our goal is to compute the minimal probability of unsafety and specify the following *maximal probabilistic safe set* with tolerance $\alpha \in (0, 1)$:

$$S^*(\alpha) := \{\boldsymbol{s} \in \mathcal{S} \mid \inf_{\pi} P_{\boldsymbol{s}}^{\text{reach}}(\pi) \leq \alpha\}.$$

This set can be used for *safety verification*: If the agent is initialized within $S^*(\alpha)$, we can guarantee safety with probability $1 - \alpha$ by carefully steering the agent; otherwise, it is impossible to do so.

We now express the probability of unsafety as an expected sum of stage-wise costs by using the technique proposed in [13]. Let $\mathbf{1}_C : \mathcal{S} \mapsto \{0, 1\}$ denote the indicator function of set $C \subseteq \mathcal{S}$ so that its value is 1 if $s \in C$; otherwise, 0. Given a sequence of states $\{s_0, \ldots, s_t\}$, we observe that

$$\prod_{k=0}^{t-1} \mathbf{1}_{\mathcal{G}^c}(s_k)\mathbf{1}_{\mathcal{G}}(s_t) = \begin{cases} 1 & \text{if } s_0, \ldots, s_{t-1} \in \mathcal{G}^c, s_t \in \mathcal{G} \\ 0 & \text{otherwise.} \end{cases}$$

It is easily seen that the sum of $\prod_{k=0}^{t-1} \mathbf{1}_{\mathcal{G}^c}(s_k)\mathbf{1}_{\mathcal{G}}(s_t)$ along the trajectory is equal to 0 if the trajectory is away from $\mathcal{G}$ and 1 if there exists at least one state s_t that is in $\mathcal{G}$. The probability of unsafety under π is then given by

$$P_{\boldsymbol{s}}^{\text{reach}}(\pi) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{T^*-1} \prod_{k=0}^{t-1} \mathbf{1}_{\mathcal{G}^c}(s_k)\mathbf{1}_{\mathcal{G}}(s_t) \mid s_0 = \boldsymbol{s}\right].$$

We introduce an auxiliary state x_t, which is an indicator of whether a trajectory $\{s_0, \cdots, s_{t-1}\}$ is fully safe or not. It is defined as

$$x_0 = 1, \quad x_t = \prod_{k=0}^{t-1} \mathbf{1}_{\mathcal{G}^c}(s_k), \quad t \geq 1.$$

Since $x_{t+1} = x_t \mathbf{1}_{\mathcal{G}^c}(s_t)$, x_{t+1} depends solely on (s_t, x_t) and a_t, so the Markov property holds with respect to the state pair (s_t, x_t). The problem of computing the minimal probability of unsafety can be formulated as

$$\inf_{\pi \in \Pi} P_{\boldsymbol{s}}^{\text{reach}}(\pi) = \inf_{\pi \in \Pi} \mathbb{E}^{\pi} \left[\sum_{t=0}^{T^*-1} x_t \mathbf{1}_{\mathcal{G}}(s_t) \mid (s_0, x_0) = (\boldsymbol{s}, 1) \right], \tag{3.1}$$

which is in the form of the standard optimal control problem. Let $V^* : \mathcal{S} \times \{0,1\} \to \mathbb{R}$ denote the optimal value function of this problem, that is, $V^*(\boldsymbol{s}, \boldsymbol{x}) := \inf_{\pi \in \Pi} \mathbb{E}^{\pi}[\sum_{t=0}^{T^*-1} x_t \mathbf{1}_{\mathcal{G}}(s_t) \mid (s_0, x_0) = (\boldsymbol{s}, \boldsymbol{x})]$. After computing the optimal value function, we can obtain the maximal probabilistic safe set by simple thresholding:

$$S^*(\alpha) = \{\boldsymbol{s} \in \mathcal{S} \mid V^*(\boldsymbol{s}, 1) \leq \alpha\}.$$

Note that this set is a superset of $S^{\pi}(\alpha) := \{\boldsymbol{s} \in \mathcal{S} \mid P_{\boldsymbol{s}}^{\text{reach}}(\pi) \leq \alpha\} = \{\boldsymbol{s} \in \mathcal{S} \mid V^{\pi}(\boldsymbol{s}, 1) \leq \alpha\}$ for any Markov policy π, where $V^{\pi} : \mathcal{S} \times \{0,1\}$ denotes the value function of π defined by $V^{\pi}(\boldsymbol{s}, \boldsymbol{x}) := \mathbb{E}^{\pi}[\sum_{t=0}^{T^*-1} x_t \mathbf{1}_{\mathcal{G}}(s_t) \mid (s_0, x_0) = (\boldsymbol{s}, \boldsymbol{x})]$. To distinguish $S^{\pi}(\alpha)$ from $S^*(\alpha)$, we refer to the former as the (probabilistic) safe set under π.

3.2 Safe Reinforcement Learning

Our goal is to compute the minimal probability of unsafety and the maximal probabilistic safe set without the knowledge of state transition probabilities in a *safety-preserving* manner. We propose an RL algorithm that guarantees the safety of the agent during the learning process for safety specification. More

specifically, the sequence $\{\pi_k\}_{k=0,1,\ldots}$ generated by the proposed RL algorithm satisfies

$$P_{\boldsymbol{s}}^{\mathrm{reach}}(\pi_{k+1}) \leq \alpha \quad \forall \boldsymbol{s} \in S^{\pi_k}(\alpha) \tag{3.2}$$

for $k = 0, 1, \ldots$. This constraint ensures that

$$S^{\pi_k}(\alpha) \subseteq S^{\pi_{k+1}}(\alpha),$$

that is, the probabilistic safe set (given α) monotonically expands. We also use the constraint (3.2) to perform *safe exploration* to collect sample data by preserving safety in a probabilistic manner.

Chapter 4

Lyapunov-Based Safe Reinforcement Learning for Safety Specification

To determine the set of safe policies that satisfy (3.2), we adopt the Lyapunov function proposed in [16] and enhance the approach to incentivize the agent to explore the state space efficiently.

Throughout the chapter, we assume that every terminal state lies in $S^*(\alpha)$ and that, at all events, an agent arrives at a terminal state in a finite period. Thus, there exists an integer m such that $\mathbb{P}^\pi(s_m \in \mathcal{S}_{\text{term}}; s_0 = \boldsymbol{s}) > 0 \ \forall \boldsymbol{s} \in \mathcal{S}, \forall \pi \in \Pi$. In Chapter 4.1 and 4.2, the state space $\mathcal{S}$ and the action space $\mathcal{A}$ are assumed to be finite. This assumption will be relaxed when discussing the deep RL version in Chapter 4.3.

Let $\mathcal{T}_d^\pi$ denote the stationary Bellman operator for the cost function $d(\boldsymbol{s}, \boldsymbol{x}) := \boldsymbol{x}\mathbf{1}_{\mathcal{G}}(\boldsymbol{s})$

$$(\mathcal{T}_d^\pi V)(\boldsymbol{s}, \boldsymbol{x}) := d(\boldsymbol{s}, \boldsymbol{x}) + \sum_{\boldsymbol{a} \in A} \pi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{x}) \sum_{\boldsymbol{s}' \in S} p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) V(\boldsymbol{s}', \boldsymbol{x}\mathbf{1}_{\mathcal{G}^c}(\boldsymbol{s}))$$

for all $(\boldsymbol{s}, \boldsymbol{x}) \in \mathcal{S}' \times \{0,1\}$, and

$$(\mathcal{T}_d^\pi V)(\boldsymbol{s}, \boldsymbol{x}) := 0$$

for all $(\boldsymbol{s}, \boldsymbol{x}) \in \mathcal{S}_{\text{term}} \times \{0,1\}$. Note that $\mathcal{T}_d^\pi$ is an m-stage contraction with respect to $\|\cdot\|_\infty$ for all $(\boldsymbol{s}, \boldsymbol{x}) \in \mathcal{S}' \times \{0,1\}$.

4.1 Lyapunov Safety Specification

We adopt the following definition of Lyapunov functions, proposed in [16]:

Definition 1. *A function $L : S \times \{0,1\} \mapsto [0,1]$ is said to be a* Lyapunov function *with respect to a Markov policy π if it satisfies the following conditions:*

$$(\mathcal{T}_d^\pi L)(\boldsymbol{s}, \boldsymbol{x}) \leq L(\boldsymbol{s}, \boldsymbol{x}) \quad \forall (\boldsymbol{s}, \boldsymbol{x}) \in \mathcal{S} \times \{0,1\} \tag{4.1a}$$

$$L(\boldsymbol{s}, 1) \leq \alpha \quad \forall \boldsymbol{s} \in S_0, \tag{4.1b}$$

where S_0 is a given subset of $S^*(\alpha)$ and $d(s,x) := x\mathbf{1}_{\mathcal{G}}(s)$.

Inequalities (4.1a) and (4.1b) are called the *Lyapunov condition* and the *safety condition*, respectively. We can show that if an arbitrary policy $\tilde{\pi}$ satisfies the Lyapunov condition, then the probability of unsafety at S_0 does not exceed the threshold α. To see this, we recursively apply $\mathcal{T}_d^\pi$ on both sides of (4.1b) and use (4.1a) and the monotonicity of $\mathcal{T}_d^\pi$ to obtain that, for any $\boldsymbol{s} \in S_0$,

$$\alpha \geq L(\boldsymbol{s}, 1) \geq (\mathcal{T}_d^{\tilde{\pi}} L)(\boldsymbol{s}, 1) \geq ((\mathcal{T}_d^{\tilde{\pi}})^2 L)(\boldsymbol{s}, 1) \geq \cdots . \tag{4.2}$$

has a unique fixed point, which corresponds to the probability of unsafety.

Due to the m-stage contraction property, $\left(\mathcal{T}_d^{\tilde{\pi}}\right)^m$ has a unique fixed point that corresponds to the probability of unsafety, $P_{\boldsymbol{s}}^{\text{reach}}(\tilde{\pi}) = V^{\tilde{\pi}}(\boldsymbol{s}, 1)$, under $\tilde{\pi}$. Therefore, by the Banach fixed point theorem, we have

$$\alpha \geq \lim_{k \to \infty} \left((\mathcal{T}_d^{\tilde{\pi}})^{km} L \right) (\boldsymbol{s}, 1) = V^{\tilde{\pi}}(\boldsymbol{s}, 1) \quad \forall \boldsymbol{s} \in S_0. \tag{4.3}$$

Given a Lyapunov function L, consider the set $\{\tilde{\pi} \mid (\mathcal{T}_d^{\tilde{\pi}} L)(\boldsymbol{s}, 1) \leq \alpha \ \forall \boldsymbol{s} \in S_0\}$. Then, any policy $\tilde{\pi}$ in this set satisfies the probabilistic safety condition $P_{\boldsymbol{s}}^{\text{reach}}(\tilde{\pi}) \leq \alpha$ for all $\boldsymbol{s} \in S_0$ by (4.3). Thus, when S_0 is chosen as $S^{\pi_k}(\alpha)$, the safety constraint (3.2) is satisfied. This set of safe policies is called the *L-induced policy set*.

We can now introduce the Lyapunov safety specification method. For iteration k, we construct the Lyapunov function L_k by using the current policy π_k and update the policy to π_{k+1} taken from the L_k-induced policy set. Specifically, we set

$$L_k(\boldsymbol{s}, \boldsymbol{x}) := \mathbb{E}^{\pi_k}\left[\sum_{t=0}^{T^*-1} (d + \epsilon_k)(s_t, x_t) \mid (s_0, x_0) = (\boldsymbol{s}, \boldsymbol{x})\right],$$

where $\epsilon_k : \mathcal{S} \times \{0, 1\} \mapsto \mathbb{R}_{\geq 0}$ is an auxiliary cost function. Following the cost-shaping method of [16], we define the auxiliary cost as the function

$$\epsilon_k(\boldsymbol{x}) := \boldsymbol{x} \cdot \min_{\boldsymbol{s} \in S_0} \frac{\alpha - V^{\pi_k}(\boldsymbol{s}, 1)}{T^{\pi_k}(\boldsymbol{s}, 1)},$$

where $T^{\pi_k}(\boldsymbol{s}, \boldsymbol{x})$ is the expected time for an agent to reach $\mathcal{G}$ or $\mathcal{S}_{\text{term}}$ the first time under policy π_k and initial state $(\boldsymbol{s}, \boldsymbol{x})$. We refer to $T^{\pi_k}(\boldsymbol{s}, 1)$ as the *first-hitting time* for the rest of this article. It is straightforward to check that the Lyapunov condition (4.1a) is satisfied with L_k. Furthermore, the function L_k satisfies the safety condition (4.1b) because, for all $\boldsymbol{s} \in S_0$,

$$\begin{aligned} L_k(\boldsymbol{s}, 1) &\leq V^{\pi_k}(\boldsymbol{s}, 1) + \epsilon_k(1) T^{\pi_k}(\boldsymbol{s}, 1) \\ &\leq V^{\pi_k}(\boldsymbol{s}, 1) + T^{\pi_k}(\boldsymbol{s}, 1) \cdot \frac{\alpha - V^{\pi_k}(\boldsymbol{s}, 1)}{T^{\pi_k}(\boldsymbol{s}, 1)} \leq \alpha. \end{aligned}$$

Therefore, L_k is a Lyapunov function.

In the policy improvement step, we select π_{k+1} from the L_k-induced policy set so the updated policy is both safe and has an expanded probabilistic safe set.

Proposition 1. *Suppose that* $\pi_{k+1} \in \{\pi \mid (\mathcal{T}_d^{\pi} L)(\boldsymbol{s}, 1) \leq \alpha \ \forall \boldsymbol{s} \in S^{\pi_k}(\alpha)\}$. *Then, we have*

$$P_{\boldsymbol{s}}^{\mathrm{reach}}(\pi_{k+1}) \leq \alpha \quad \forall \boldsymbol{s} \in S^{\pi_k}(\alpha),$$

and $S^{\pi_k}(\alpha) \subseteq S^{\pi_{k+1}}(\alpha)$.

Proof. The probability of safety of π_{k+1} follows from (4.3). This also implies that for an arbitrary $\boldsymbol{s} \in S^{\pi_k}(\alpha)$, we have $\boldsymbol{s} \in S^{\pi_{k+1}}(\alpha)$. Therefore, the result follows. □

To achieve the minimal probability of unsafety, we choose π_{k+1} as the "safest" one in the L_k-induced policy, that is,

$$\pi_{k+1}(\cdot|\boldsymbol{s}) \in \arg\min_{\pi(\cdot|\boldsymbol{s})} \{(\mathcal{T}_d^{\pi} V_k)(\boldsymbol{s}, 1) \mid (\mathcal{T}_d^{\pi} L_k)(\boldsymbol{s}, 1) \leq L_k(\boldsymbol{s}, 1)\}. \tag{4.4}$$

Note that the value of Lyapunov function is 0 at $\boldsymbol{x} = 0$, so we need not compute a policy for $\boldsymbol{x} = 0$ ·

As the MDP model is unknown, we approximate the value function of a policy using sample trajectories. We also use Q-learning to obtain a reliable estimate of state-action value functions. Let Q_V and Q_T denote the Q-functions for the probability of unsafety and a first-hitting time, respectively. Given (s_t, a_t, s_{t+1}) obtained by executing π_k, the Q-functions are updated as follows:

$$\begin{aligned}
Q_V(s_t, a_t) &\leftarrow \mathbf{1}_{\mathcal{G}}(s_t) + \mathbf{1}_{\mathcal{G}^c}(s_t)\left[(1-\tau_l)Q_V(s_t, a_t) + \tau_l \sum_{\boldsymbol{a} \in A} \pi_k(\boldsymbol{a}|s_{t+1}) Q_V(s_{t+1}, \boldsymbol{a})\right] \\
Q_T(s_t, a_t) &\leftarrow \mathbf{1}_{\mathcal{G}^c}(s_t)\left[\tau_l\left(1 + \sum_{\boldsymbol{a} \in A} \pi_k(\boldsymbol{a}|s_{t+1}) Q_T(s_{t+1}, \boldsymbol{a})\right) + (1-\tau_l) Q_T(s_t, a_t)\right],
\end{aligned} \tag{4.5}$$

where $\tau_l(\boldsymbol{s}, \boldsymbol{a})$ is the learning rate satisfying $\sum_l \tau_l(\boldsymbol{s}, \boldsymbol{a}) = \infty$ and $\sum_l \tau_l^2(\boldsymbol{s}, \boldsymbol{a}) < \infty$. We can also rewrite (4.4) as the following linear program associated with

Q-functions:

$$\begin{aligned} \min_{\pi(\cdot|\boldsymbol{s})} \ & \sum_{\boldsymbol{a}\in\mathcal{A}} \pi(\boldsymbol{a}|\boldsymbol{s}) Q_{V,k}(\boldsymbol{s},\boldsymbol{a}) \\ \text{s.t.} \ & \sum_{\boldsymbol{a}\in\mathcal{A}} Q_{L,k}(\boldsymbol{s},\boldsymbol{a})(\pi(\boldsymbol{a}|\boldsymbol{s}) - \pi_k(\boldsymbol{a}|\boldsymbol{s})) \leq \epsilon_k, \end{aligned} \tag{4.6}$$

7-10where $Q_{L,k}$ is the Q-value of Lyapunov function given by $Q_{L,k}(\boldsymbol{s},\boldsymbol{a}) = Q_{V,k}(\boldsymbol{s},\boldsymbol{a}) + \epsilon_k(1)Q_{T,k}(\boldsymbol{s},\boldsymbol{a})$ and ϵ_k is the shortened expression of $\epsilon_k(1)$. The policy $\pi_{k+1}(\cdot|\boldsymbol{s})$ is then updated as the optimal solution of the linear program (4.6).

Combining the policy evaluation and the policy improvement steps of Q-functions, we construct the *Lyapunov safety specification* (LSS) as described in Algorithm 1. The convergence property of Q-learning in finite-state, finite-action space is well studied in [26], so we omit the theoretical details here. Under the standard convergence condition for Q-learning, the algorithm obtains a sequence of policies that satisfy Proposition 1.

4.2 Efficient Safe Exploration

In this section, we develop a novel for safe exploration to efficiently solve a probabilistic safety specification problem. We can utilize the Lyapunov constraint to construct a policy that takes potentially dangerous actions with adequate probability and thus assures safe navigation.

We take our motivation from the discovery that if a state is falsely assumed to have a high probability of unsafety, it is unlikely to correct the misconception without taking exploratory actions. Consider the table of Q-value estimates used in the LSS algorithm. The Q-learning agent is initiated from the blank slate, so it is a safe choice to assume that all unvisited states evolve into the target set with high probability. As a result, the safe policy derived from the algorithm tends to confine an agent inside the current safe set. With enough

Algorithm 1: LSS Q-Learning

Require: Tolerance for unsafety $\alpha \in (0, 1)$,
baseline policy π_{base};

Set initial policy π_0 as π_{base};
for each iteration k **do**
  for each environment step l **do**
    $a_t \sim \pi_k(\cdot|s_t)$
    Get $s_{t+1} \sim p(\cdot|s_t, a_t)$ and $\mathbf{1}_{\mathcal{G}}(s_t)$;
    Update $Q_V(s_t, a_t)$, $Q_T(s_t, a_t)$ as (4.5);
    Reset the environment if $\mathbf{1}_{\mathcal{G}}(s_t) = 1$;
  end for
  Update $\pi_{k+1}(\cdot|\boldsymbol{s})$ by solving (4.6) for each $\boldsymbol{s}$;
end for

time, the Q-value table becomes accurate at all states, but this is unattainable in practice. Therefore, it is crucial to explore the unidentified states, and this process involves visiting the exterior of the safe set.

In this regard, we choose the exploratory policy to be the most aggressive among the set of policies that guarantee safety in the safe set. Conversely, the probabilistic safety of the exploratory policy in the safe set is marginally greater than the tolerance. As there is no element $\mathcal{G}$ in $S^{\pi_s}(\alpha)$, such a policy is likely to bring an agent outside the safe set. The exploratory policy is efficient if used with an experience replay, the state distribution of which may diverge from the true distribution due to the scarcity of samples obtained in the exterior of the safe set. Our exploratory policy can mitigate the approximation error due to the discrepancy.

To illustrate our idea, we show a one-dimensional (1D) grid world consisting

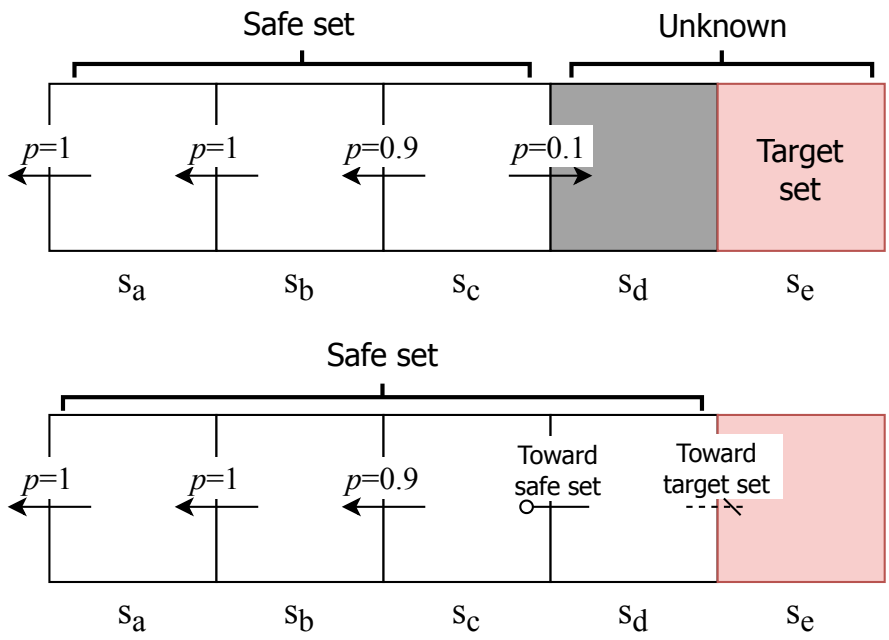


Figure 4.1 An example of safe exploration on a one-dimensional grid world. The confidence level is set to 0.9. Boxes represent states, and arrows toward the left or right symbolize the policies at each state. Unexamined states are shaded; the gray one is not in the target set, but it is considered unsafe. Choosing the policy at $\boldsymbol{s}_c$ allows an agent to explore toward $\boldsymbol{s}_d$ (top). As the RL agent successfully returns to the safe set after visiting $\boldsymbol{s}_d$ with high probability, $\boldsymbol{s}_d$ is added to the safe set (bottom).

of five states $\boldsymbol{s}_a, \ldots, \boldsymbol{s}_e$ and two actions (left, right) as in Fig. 4.1. We know from experience that moving to the left at $\boldsymbol{s}_a, \ldots, \boldsymbol{s}_c$ guarantees 100% safety. The states $\boldsymbol{s}_d$ and $\boldsymbol{s}_e$ are not visited yet, so the probabilities of unsafety at those states are 1. Suppose the agent is in $\boldsymbol{s}_c$ and chooses to move left or right with probability $(1-\alpha, \alpha)$. The probability of unsafety of π is then no more than α because an agent never reaches $\boldsymbol{s}_d$ or $\boldsymbol{s}_e$ with probability $1-\alpha$. Also, if an agent

Algorithm 2: ESS Q-Learning

Require: Tolerance for unsafety $\alpha \in (0, 1)$,

baseline policy π_{base};

Set $\pi_{s,0} \leftarrow \pi_{\text{base}}$ and $\pi_{e,0} \leftarrow \pi_{\text{base}}$;

for each iteration k **do**

for each environment step l **do**

$a_t \sim \pi_{e,k}(\cdot|s_t)$;

Get $s_{t+1} \sim p(\cdot|s_t, a_t)$ and $\mathbf{1}_{\mathcal{G}}(s_t)$;

Update $Q_V(s_t, a_t)$, $Q_T(s_t, a_t)$ as (4.5);

Reset environment if $\mathbf{1}_{\mathcal{G}}(s_t) = 1$;

end for

Set $\pi_{s,k+1}(\cdot|\boldsymbol{s})$ by solving (4.6) for each $\boldsymbol{s}$;

Set $\pi_{e,k+1}(\cdot|\boldsymbol{s})$ by solving (4.8) for each $\boldsymbol{s}$;

end for

successfully reaches $\boldsymbol{s}_d$ or $\boldsymbol{s}_e$ and returns safely, we obtain an accurate estimate of the probability of unsafety and expand the safe set.

A policy suitable for exploration is not usually the safest policy; therefore, we separate the *exploratory policy* π_e from the policy that constructs the safe set, which is denoted by the *safety specification-policy* (SS-policy) π_s. Unlike the SS-policy, the exploratory policy drives an agent around the boundary of the safe set. To construct π_e in a formal way, we exploit a given π_s and the Lyapunov function L defined as in Chapter 4.1. First, consider the following policy optimization problem:

$$\begin{aligned} &\max_{\pi \in \Pi} \ V^{\pi}(s_0, 1) \\ &\quad \text{s.t. } (\mathcal{T}_d^{\pi} L)(\boldsymbol{s}, \boldsymbol{x}) \leq L(\boldsymbol{s}, \boldsymbol{x}) \quad \forall (\boldsymbol{s}, \boldsymbol{x}) \in \mathcal{S} \times \{0, 1\}, \end{aligned} \tag{4.7}$$

where s_0 is an initial state. Note that this is the auxiliary problem merely to

construct the exploratory policy with no connection to the original problem (3.1). As stated above, the exploratory policy should preserve safety confidence in the safe set under the SS-policy, that is, $V^{\pi_e}(\boldsymbol{s}, 1) \leq \alpha,\ \forall \boldsymbol{s} \in S^{\pi_s}(\alpha)$. The solution of (4.7) satisfies this condition because of the Lyapunov constraint, but it can be suboptimal because the constraint in (4.7) is stronger than the original. However, by using the Lyapunov constraints, we can enjoy the benefit of using dynamic programming to solve (4.7).

Proposition 2. *Let L be the Lyapunov function stated in* (4.7)*. An optimal solution of* (4.7) *can be obtained by the value iteration using the Bellman operator*

$$\begin{aligned}&(\mathcal{T}_{\text{exp}}V)(\boldsymbol{s}, \boldsymbol{x})\\ &:= \max_{\pi(\cdot|\boldsymbol{s})}\{(\mathcal{T}_d^{\pi}V)(\boldsymbol{s}, \boldsymbol{x}) \mid (\mathcal{T}_d^{\pi}L)(\boldsymbol{s}, \boldsymbol{x}) \leq L(\boldsymbol{s}, \boldsymbol{x})\}.\end{aligned}$$

Specifically, the value function that satisfies $\mathcal{T}_{\text{exp}}V = V$ is the probability of unsafety under such a policy.

Proof. The operator $\mathcal{T}_{\text{exp}}$ is a special form of the safe Bellman operator defined in [16], which is a monotone contraction mapping by Proposition 3 in [16]. Thus, there exists a unique fixed point of $\mathcal{T}_{\text{exp}}$. By the definition of the operator, the fixed point corresponds to the policy and solves problem 4.7. □

As Proposition 2 certifies, we can perform the Bellman operation on V^{π_s} iteratively to obtain π_e, which is the solution of (4.7). However, in the RL domain, it is difficult to reproduce the whole dynamic programming procedure, since each Bellman operation corresponds to a time-consuming Q-value computation. We thus apply the Bellman operation once to obtain $\pi_e(\cdot|\boldsymbol{s})$ at iteration number k as

$$\arg\max_{\pi(\cdot|\boldsymbol{s})}\{(\mathcal{T}_d^{\pi}V_k)(\boldsymbol{s}, 1) \mid (\mathcal{T}_d^{\pi}L_k)(\boldsymbol{s}, \boldsymbol{x}) \leq L_k(\boldsymbol{s}, \boldsymbol{x})\}. \tag{4.8}$$

To sum up, we add an exploratory policy to LSS to obtain the *exploratory LSS* (ESS), as Algorithm 2.

4.3 Deep RL Implementation

Each policy improvement stages in Algorithm 1 or 2 solves a linear program. This operation is not straightforward for nontabular implementations. Thus, we provide adaptations of the LSS and ESS for parametrized policies, such as neural networks. To apply our approach to high-dimensional environments in this section, we assume that the state and action spaces are continuous, which is the general setup in policy gradient (PG) algorithms. Suppose a generic policy is parameterized with θ, and we rewrite the policy improvement step of the LSS as

$$\begin{aligned} &\max_{\theta} \int_{\mathcal{A}} -Q_V(\boldsymbol{s},\boldsymbol{a})\pi_\theta(\boldsymbol{a}|\boldsymbol{s})\,\mathrm{d}\boldsymbol{a} \quad \text{subject to} \\ &\int_{\mathcal{A}} Q_L(\boldsymbol{s},\boldsymbol{a})\left(\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) - \pi_s(\boldsymbol{a}|\boldsymbol{s})\right)\mathrm{d}\boldsymbol{a} \le \epsilon \quad \forall \boldsymbol{s} \in \mathcal{S}, \end{aligned} \tag{4.9}$$

where π_s is the current SS-policy and Q_V, Q_L, and ϵ are the values defined as the previous chapter with respect to π_s.

We use Lagrangian relaxation [27] to form an unconstrained problem. Ignoring the constraints, the PG minimizes a single objective $\mathbb{E}_{s,a\sim\pi}[Q_V(s,a)]$. The Lyapunov condition is state-wise, so the number of constraints is the same as $|\mathcal{S}|$. We can replace the constraints with a single one

$$\max_{\boldsymbol{s}\in\mathcal{S}} \int_{\mathcal{A}} Q_L(\boldsymbol{s},\boldsymbol{a})\left(\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) - \pi_s(\boldsymbol{a}|\boldsymbol{s})\right)\mathrm{d}\boldsymbol{a} - \epsilon \le 0.$$

However, one drawback of this formulation is that the Lagrangian multiplier of the max-constraint places excessive weight on the constraint. In practice, the LHS of this max-constraint is likely greater than 0 due to the parameterization errors, resulting in the monotonic increase of the Lagrangian multiplier

throughout learning. Therefore, we adopt state-dependent Lagrangian multipliers to have

$$\begin{aligned}&\min_{\lambda\geq 0}\max_{\theta}\mathbb{E}_{s\sim\rho_\theta}\big[\mathbb{E}_{a\sim\pi_\theta}[-Q_V(s,a)]\\&-\lambda(s)\left(\mathbb{E}_{a\sim\pi_\theta}[Q_L(s,a)]-\mathbb{E}_{a\sim\pi_s}[Q_L(s,a)]-\epsilon\right)\big],\end{aligned}\tag{4.10}$$

where $\lambda(s)$ is the Lagrangian multiplier at state s, and ρ_θ is the discounted state-visiting probability of π_θ. We can assume that nearby states have similar $\lambda(s)$. Thus, we can parameterize $\lambda(s)$ as a critic model, as in [28]. Throughout this section, we represent ω as the parameter of λ.

Our goal is to find the saddle point of (4.10), which is a feasible solution of the original problem (4.9). We apply the gradient descent (ascent) to optimize θ and ω. The Q-values that comprise the Lagrangian are, by definition, the functions of the policy parameter θ, but since we incorporate the actor-critic framework, the Q-functions are approximated with critics independent of θ. In this regard, we obtain the update rules for the safety specification-actor (SS-actor) and the Lagrangian multiplier associated with it as follows:

$$\theta_s \leftarrow \theta_s - \eta_\theta\nabla_\theta\left(Q_V(s_t,a_t)+\lambda_{\omega_s}(s_t)Q_L(s_t,a_t)\right),\tag{4.11a}$$

$$\omega_s \leftarrow \omega_s + \eta_\omega\nabla_\omega\lambda_{\omega_s}(s_t)\big(Q_L(s_t,a_t)-\epsilon-Q_L(s_t,a_{\text{old},t})\big),\tag{4.11b}$$

where $a_t \sim \pi_{\theta_s}(s_t)$ and $a_{\text{old},t}$ denotes the sampled action from the policy parametrized with the old θ_s.

We apply the same approach to improve the exploratory actor. The unconstrained problem is similar to (4.10) except for the opposite sign of the primal objective, so we have

$$\theta_e \leftarrow \theta_e + \eta_\theta\nabla_\theta\left(Q_V(s_t,a_t)-\lambda_{\omega_e}(s_t)Q_L(s_t,a_t)\right)\tag{4.12a}$$

$$\omega_e \leftarrow \omega_e + \eta_\omega\nabla_\omega\lambda_{\omega_e}(s_t)\big(Q_L(s_t,a_{\text{exp},t})-\epsilon-Q_L(s_t,a_t)\big),\tag{4.12b}$$

where $a_{\text{exp},t} \sim \pi_{\theta_e}(s_t)$, $a_t \sim \pi_{\theta_s}(s_t)$.

Besides, critic parameters are optimized to minimize the Bellman residual. The scheme is analogous to the Q-learning version, as in (4.5), but in this case, we express the discount factor γ. Recall that the Lyapunov Q-function is a weighted sum of the two Q-functions Q_V and Q_T, one for a probability of unsafety and the other for a first-hitting time, respectively. Letting ϕ and ψ represent the parameters of Q_V and Q_T, the targets for the critics Q_ϕ and Q_ψ are defined as

$$
\begin{aligned}
y_V &:= \mathbf{1}_{\mathcal{G}}(s_t) + \mathbf{1}_{\mathcal{G}^c}(s_t)\gamma Q_{\phi'}(s_{t+1}, a_{t+1}) \\
y_T &:= \mathbf{1}_{\mathcal{G}^c}(s_t)(1 + \gamma Q_{\psi'}(s_{t+1}, a_{t+1})),
\end{aligned}
$$

where a_{t+1} is the action sampled from $\pi_{\theta'_s}(s_{t+1})$. The proposed actor-critic algorithm is summarized in Algorithm 3.

In our experiments, we use the double Q-learning technique in [29] to prevent the target y_V from being overly greater than the true probability of unsafety. In this case, two critics have independent weights ϕ_1, ϕ_2, and two target critics pertained to the respective critics. That is, $Q_{\phi'}(s_{t+1}, a_{t+1})$ in y_V is replaced with $\min_{j=1,2} Q_{\phi'_j}(s_{t+1}, a_{t+1})$, where $a_{t+1} \sim \pi_{\theta'_s}(s_{t+1})$. Moreover, we adjust the experience replay to alleviate errors in Q_V. Catastrophic forgetting is the primary concern, since the target set should be precisely specified to obtain safe policies. We fix the ratio of safe samples (*i.e.*, $s_t \notin \mathcal{G}$) and unsafe samples (*i.e.*, $s_t \in \mathcal{G}$) in a minibatch so that the value of Q_V is 1 in the identified states of the target set. We explain the ancilliary techniques in Chapter 5.2.

Algorithm 3: Actor-critic LSS (ESS)

Require: Tolerance for unsafety $\alpha \in (0, 1)$;

Initialize SS-actor/critics $\pi_{\theta_s}, Q_\phi, Q_\psi$ and Lagrangian multiplier λ_{ω_s};

if ESS **then**

Initialize exploratory actor π_{θ_e} and Lagrangian multiplier λ_{ω_e};

end if

Initialize target networks: $\theta'_s \leftarrow \theta_s$, $\psi' \leftarrow \psi$, $\phi' \leftarrow \phi$;

for each iteration t **do**

for each environment step **do**

$a_t \sim \pi_{\theta_s}(\cdot|s_t)$ (Use π_{θ_e} if ESS);

$s_{t+1} \sim p(\cdot|s_t, a_t)$;

$\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, \mathbf{1}_{\mathcal{G}}(s_t), s_{t+1}\}$;

Reset environment if $\mathbf{1}_{\mathcal{G}}(s_t) = 1$;

end for

for each gradient step **do**

Update ϕ by minimizing $(y_V - Q_\phi(s_t, a_t))^2$;

Update ψ by minimizing $(y_T - Q_\psi(s_t, a_t))^2$;

Update θ_s as the solution of (4.11a);

Update ω_s as the solution of (4.11b);

if ESS **then**

Update θ_e as the solution of (4.12a);

Update ω_e as the solution of (4.12b);

end if

end for

Soft target update for SS-actor/critic: $\theta'_s \leftarrow \tau\theta_s + (1-\tau)\theta'_s$,

$\phi' \leftarrow \tau\phi + (1-\tau)\phi'$, $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$;

end for

Chapter 5

Simulation Studies

In this chapter, we demonstrate our safe learning and safety specification methods using simulated control tasks. We test the validity of our approach in a simple double integrator and further verify our deep RL algorithms with the high-dimensional dynamic system introduced in [30], both of which have a tuple of positions and velocities as a state. To make the environments congruous with our problem setting, a target set is defined as the exterior of a certain bounded region of the state space, a setup that enables the implementation of tabular Q-learning. The description of environments, including the definition of target sets, can be found in Chapter 5.3.2.

In Chapter 4, we stated the theoretical guarantees as follows. First, Lyapunov-based methods obtain a subset of $S^*(\alpha)$. Second, the improved safe set includes the current safe set. Third, the agent ensures safety while running in the environment if the initial state is safe. However, in practice, these guarantees cannot be strictly satisfied, since we determine a safe set with the approximated probability of unsafety. To distinguish the obtainable safe set from the ideal one

derived from the true MDP, we represent the estimate of the safe set under π as

$$\hat{S}^{\pi}(\alpha) := \{\boldsymbol{s} \in \mathcal{S} : \pi_s(\cdot|\boldsymbol{s})\hat{Q}_V(\boldsymbol{s}, \cdot) \leq \alpha\}.$$

We introduce two metrics to quantify how close well-trained RL agents are to such guarantees. Regarding the accuracy of safety specification, we inspect if a safe set contains the elements of $S^*(\alpha)$ and if it does not include the unreliable states in $S^*(\alpha)^c$. We thus consider the *ratio of correct specification*

$$r_{\mathrm{c}} = \frac{|\hat{S}^{\pi}(\alpha) \cap S^*(\alpha)|}{|S^*(\alpha)|},$$

and the *ratio of false-positive specification*

$$r_{\mathrm{fp}} = \frac{|\hat{S}^{\pi}(\alpha) \cap S^*(\alpha)^c|}{|\mathcal{S}|}.$$

We also verify safe exploration throughout learning by tracking the proportion of safely terminated episodes among the 100 latest episodes, which is denoted by the *average episode safety* (AES). A trajectory is considered safe if an agent reaches terminal states without visiting $\mathcal{G}$ or stays in $\mathcal{G}^c$ until the time limit.

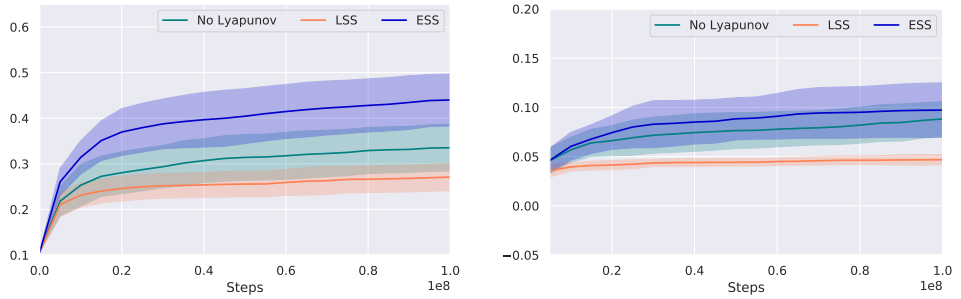
Throughout our experiments, we set $\alpha = 0.2$, so AES should be no less than 0.8 to guarantee safe navigation. We also improve learning speed by introducing a discount factor $\gamma < 1$, which is equivalent to $p(s_{t+1} \in \mathcal{S}_{\mathrm{term}}|s_t, a_t)$. To observe the impact of Lyapunov constraints, we set unmodified RL methods as baseline, *i.e.* the agents trained to minimize the expected sum of $x_t \mathbf{1}_{\mathcal{G}}(s_t)$. The other safe RL algorithms can be compared with ours, but the comparison would be unfair in terms of the safe learning condition (3.2); thus, we focus on the validity of our approach.

5.1 Tabular Q-Learning

First, we evaluate our Lyapunov-based safety specification methods with tabular implementations. For tabular Q-learning, we discretize a continuous action $a = (a_1, \cdots, a_{\dim \mathcal{A}})$ into partitions of $A_1, \cdots, A_{\dim \mathcal{A}}$ equal intervals for each element. In other words, applying the nth action for a_m is interpreted as $a = (a_{m,\max} - a_{m,\min})\frac{n}{A_m - 1} + a_{m,\min}$. Likewise, state space is represented as a finite-dimensional grid. Based on the MDP quantized as above, the true probability of safety is computed via dynamic programming.

We use a double integrator environment to test the tabular cases. To reduce the size of $S^*(\alpha)$, we modify the integrator to perturb the input acceleration with a certain probability. We compare LSS, ESS, and a baseline Q-learning with no extra techniques to shield unsafe actions. We initialize the Q-function tables with random values uniformly sampled from the interval $[0.99, 1]$; that is, the probability of unsafety is estimated as nearly 1 in all states. Therefore, in the tabular setting we impose the assumption that all unvisited states have the probabilistic safeties lower than the threshold. We then perform the policy improvement 10^2 times, each of which comes after 10^6 environment steps.

Fig. 5.1 summarizes the specification result averaged across 20 random seeds. Both LSS and ESS show monotonic improvement of the approximated safe set $\hat{S}^\pi(\alpha)$. Notably, we find evidence of ESS taking advantage of exploration. The r_c of ESS increase faster than those of LSS or the baseline, while the excess of r_fp of ESS is negligible. The average value of r_c is 44% for ESS, surpassing the baseline of 34%. The effect of ESS culminates at the beginning of the learning process then dwindles because the boundary of $\hat{S}^\pi(\alpha)$ becomes unlikely to reach as the set inflates, so the chance of exploration decreases. Ideally, with the appropriate choice of $\gamma \approx 1$ and the learning rate, r_fp is nearly



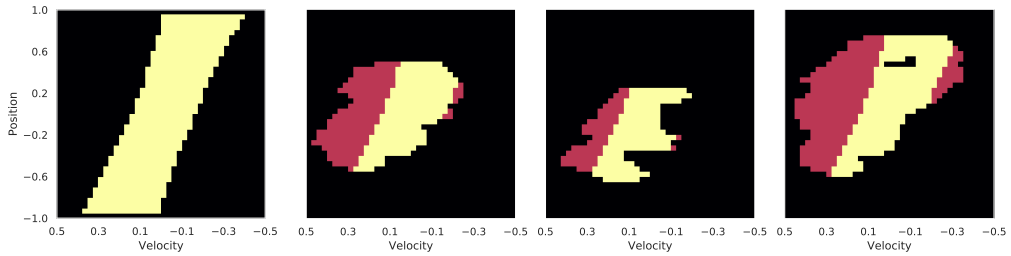
(a) Integrator, r_{c}

(b) Integrator, r_{fp}

Figure 5.1 Safety specification via tabular Q-learning tested on the double integrator. The solid line denotes the average, and the shaded area indicates the confidence interval of 20 random seeds. The baseline, LSS, and ESS are denoted by teal, orange, and blue, respectively.

0. We skip the AES in Fig. 5.1, since no agent lacks safety confidence. However, the AES might decline without the limit, since an episode is configured to terminate after 200 steps, which restricts the chance of reaching the target set.

We illustrate the safety analysis results of respective methods and the ground-truth probabilistic safe set in Fig. 5.2. Each approximated safe set is established from the Q-learning table of an agent with the highest rate of correct specification among the 20 random seeds analyzed in Fig. 5.1. A grid map represents the whole non-target set except for the grid points on the sides, and the approximated safe set is the set of red and yellow points. The size of $\hat{S}^{\pi}(\alpha)$ for ESS is notably larger than that of the baseline or LSS in the cases of both correctly specified states (yellow) and misclassified states (red). However, the false-positive in the safe set estimated by ESS is hardly due to the ESS method but comes from a universal limitation of tabular Q-learning. This can be explained from the observation that the ratio of misclassified states over the whole



(a) Ground truth (b) Baseline (c) LSS (d) ESS

Figure 5.2 Safe sets for the integrator problem with $\alpha = 0.2$. Each grid point denotes a state (position, velocity). The ground truth $S^*(\alpha)$ is denoted by yellow in (a). The other figures show the safe set estimated by (b) the baseline, (c) LSS, and (d) ESS. The shaded region represents $\hat{S}^\pi(\alpha)$: correctly specified states are marked yellow, and unsafe states misclassified as safe are marked red.

$\hat{S}^\pi(\alpha)$ of ESS is greater than that of the baseline only by 5%; that is, ESS does not particularly overestimate the probability of safety in unsafe states. The ESS Q-learning is expected to obtain an accurate estimate of S^* if the implementation of Q-learning is improved.

5.2 Deep RL

We present the experimental results in Algorithm 3 using a realistic robotic simulation. We demonstrate that our approach can be coupled with well-established deep RL methods to perform safety specifications efficiently in the continuous state and action space. Details about our deep RL implementation can be found in Chapter 5.3.1. We consider a *Reacher* system for safety analysis. In the Reacher, safety constraints are set on the position of the end effector (See Chapter 5.3.2 for details).

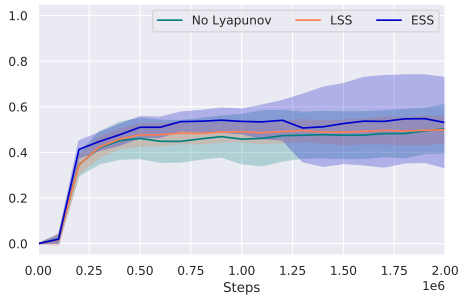
We implement the LSS and ESS actor-critic in DDPG [31], and the baseline.

For the sake of fairness, all the algorithms use the same actor network weight and the same replay memory at the start of learning. The critics are initialized randomly, but the bias value for each layer of Q_V is set to 1 so that $Q_V(\boldsymbol{s}, \boldsymbol{a}) = 1$ for almost all $(\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S} \times \mathcal{A}$. This ensures that the ratio of correct specification is 0 at the very beginning. We also optimize only the critics for the first 10^5 steps to reduce the discrepancies between critics and actors. The techniques mentioned in Chapter 4.3 are also applied: we fill 20% of each minibatch with the unsafe samples and use double Q_V networks for critic update.

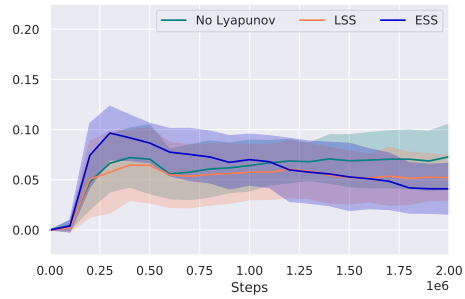
The Lyapunov-based RL agents require auxiliary cost ϵ, as in Chapter 3. For the case of a continuous state space, the safe set is not explicitly defined, so ϵ should be approximated. We first set the denominator of ϵ to $T^\pi(\boldsymbol{s}) \approx (1-\gamma)^{-1}$ to prevent it from being larger than the true value. To estimate $\min_{\boldsymbol{s} \in S^*(\alpha)}\{\alpha - V^\pi(\boldsymbol{s}, 1)\}$, we use supplementary memory that remembers the value of $\{\alpha - V^\pi(\boldsymbol{s}, 1)\}^+$ for $\boldsymbol{s}$ such that $V^\pi(\boldsymbol{s}, 1) \leq \alpha$.s When an episode is terminated, an agent computes V^π for all the states in the trajectory and find the maximum among the values that satisfy $V^\pi(\boldsymbol{s}, 1) \leq \alpha$. The memory stores the result for the 100 latest trajectories.

We also exploit the two actors of the ESS actor-critic to ensure safe operation. Since it takes time to construct a stable exploratory actor, the agent makes stochastic choices between the two actors in the early stages. The probability of an SS-actor being chosen is 1 at the first gradient step and declines linearly until it becomes 0 after the first half of the learning process. The SS-actor is also utilized as the backup policy; that is, the agent takes the action using π_s if the AES is less than the threshold $1 - \alpha$, regardless of the policy choice scheme described above. To reduce computation time, λ_{ω_s} is fixed to 0 for the ESS actor-critic.

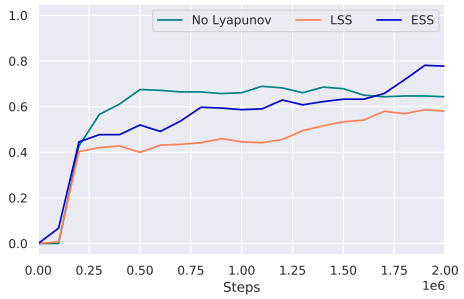
Fig. 5.3 summarizes the experimental result. We perform tests on 10 random



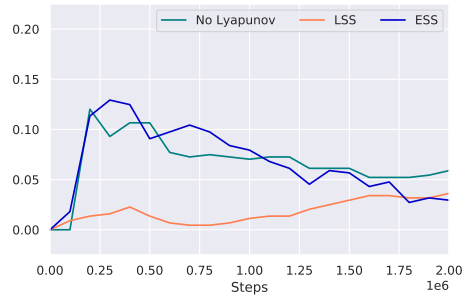
(a) Reacher, r_{c} (average)



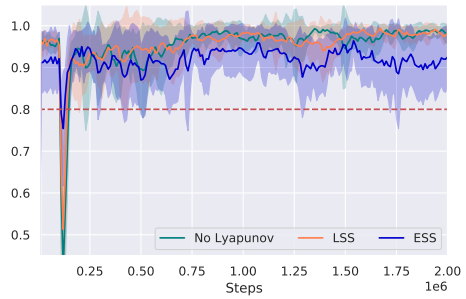
(b) Reacher, r_{fp} (average)



(c) Reacher, r_{c} (best)



(d) Reacher, r_{fp} (best)



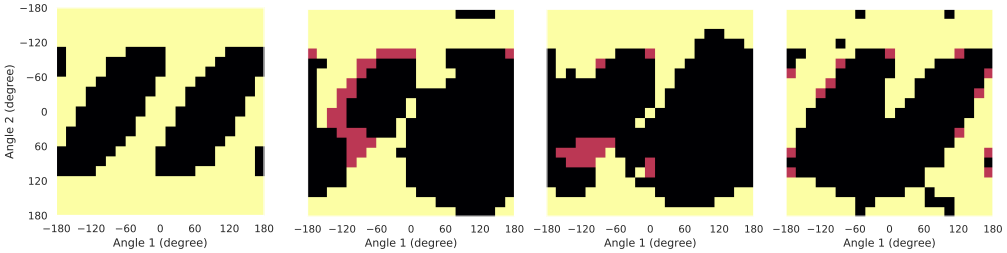
(e) Reacher, AES

Figure 5.3 Safety specification via deep RL tested on the Reacher. (a-b) are the results averaged across 10 random seeds, and (c-d) are the best results for various methods. (e) displays the average episode safety swept across all seeds. Color schemes are equivalent to Fig. 5.1.

seeds to take an average (5.3a, 5.3b) and to display the ones that attain the greatest r_{c} among various methods (5.3c, 5.3d). Comparing the average cases, the ESS actor-critic shows improvement in both specification criteria, and is noticeable for false positives. ESS consistently reduces r_{fp} except for the first 3×10^5 steps and then achieves 4.10%, while the baseline and LSS settle at 7.30% and 5.22%, respectively. The learning curves of ESS and the baseline are similar at the very start, since ESS does not regularly use the exploratory policy then. The exploratory policy in ESS supplements novel information about the states, which are normally the elements of the target set, and the safe set thus becomes more accurate. On the other hand, those of the baseline stay stagnant because the agent barely falls into an unusual trajectory with the SS-policy. Regarding LSS, we observe that the regularization term in its update rule degrades the overall performance.

As seen by the large confidence interval of ESS in Fig. 5.3a, the effect of the exploratory policy varies. ESS performs as the description in Chapter 4.2; considering the best cases, ESS attains 77.7% for the correct specification, which is 13.4% above the baseline. The exploratory policies sometimes converge fast and become indifferent to the SS-policies in terms of exploration, resulting in poor performance. Note that the difference in ESS behavior is determined by the approximation error in the critic Q_V. Although it is difficult to organize the parametrized critic, we can exploit the potential of ESS by running on multiple seeds and finding the best among them.

In Fig. 5.4, we further visualize a relevant part of the state space and the safe sets in it. Each grid map displays $\hat{S}^\pi(\alpha)$ of the agent whose r_{c} is the greatest among the 10 random seeds discussed above. The safe set obtained by ESS clearly resembles the true safe set better than the others.



(a) Ground truth (b) Baseline (c) LSS (d) ESS

Figure 5.4 Safe sets in the state space of the Reacher. Each grid point denotes a state of the end effector whose position is determined by the angles of the two joints and whose velocity is 0. Given $\alpha = 0.2$, the ground truth $S^*(\alpha)$ is denoted by yellow in (a). The other figures show the estimated safe set obtained by (b) the baseline, (c) LSS, and (d) ESS. Color schemes are equivalent to Fig. 5.2.

5.3 Experimental Setup

5.3.1 Deep RL Implementation

In this part, we provide a specific description of the deep RL agents used in our experiments. Table 5.1 displays the basic architecture of neural networks, all of which are fully connected and consist of two hidden layers with ReLU as an activation function unless it is an estimator of Q_V. The first and second hidden layers have 400 and 300 nodes, respectively. Adam optimizer [32] is used to apply gradient descent. Aside from the techniques stated in Chapter 5.2, an action is perturbed with Ornstein-Uhlenbeck noise with parameters $\mu = 0$, $\theta = 0.1$, and $\sigma = 0.05$.

Type	Output size	Activation	Learning rate
Critic	1	clamp$(0, 1)$	10^{-4}
Actor	dim(a)	tanh	10^{-5}
$\log \lambda$	1	clamp$(-10, 6)$	10^{-6}

Table 5.1 The top layers of respective networks in DDPG.

5.3.2 Environments

An environment provides a Boolean `done` signal that declares the termination of an episode strictly equivalent to $\mathbf{1}_{\mathcal{S}_{\text{term}}}$. When its value is 1, both Q_V and Q_T at that state are set to 0. If the length of an episode exceeds the time limit before arriving at a terminal state, the environment resets itself, but `done` is still 0 at that moment. Refer to Table 5.2 for the time limit and the discount factor settings.

Randomized integrator. A vanilla double integrator is a system with a 2D state (x_1, x_2) and the scalar control u. x_1 and x_2 represent the position and velocity on a 1D line, respectively. The control is an acceleration.

We add a few features to construct a safety specification problem in this environment. First, we set the terminal states as the points near the origin $(x_1, x_2) \in [-0.2, 0.2] \times [-3.75 \times 10^{-3}, 3.75 \times 10^{-3}]$. Next the target set is defined as all the states $(x_1, x_2) \notin [-1, 1] \times [-0.5, 0.5]$. Finally, we restrict admissible action to the range $[-0.5, 0.5]$, and adjust the dynamics so that the acceleration is scaled to $0.5u/|u|$ with probability $1/2$. Due to the introduction of stochastic behavior, it becomes more difficult to reach the terminal states safely than in the original environment.

Reacher. Reacher is a simulative planar 2-DOF robot with two arms attached to joints implemented with a Mujoco engine [33]. The joint of the

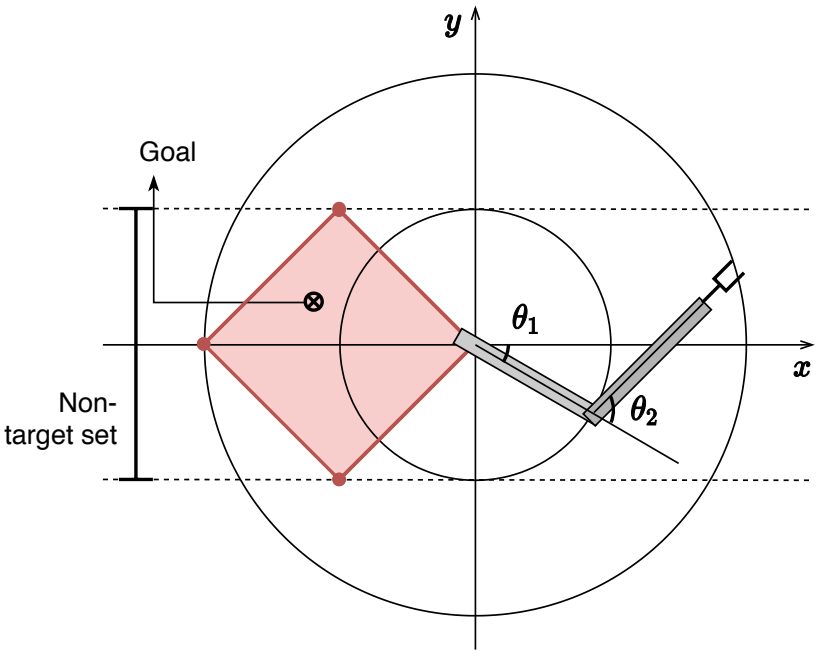


Figure 5.5 Description of the Reacher environment.

first arm is fixed on the center of the plane, and the joint of the second is connected to the movable end of the first. The objective of the robot is to touch a motionless goal point with its end effector. An observation is thus defined as a vector that contains the angular positions and the angular velocities of the joints as well as the position of the goal. The action is defined as the torques on the joints, each of which is bounded in the range $[-1, 1]$.

Let the coordinates be defined as in Fig. 5.5. Specifically, the goal is deployed randomly in the hued area $\{(x, y)||x + l| + |y| \leq l\}$, where l is the length of one arm. The exact position changes for each reset. We define the target set as $\{(x, y)||y| > l\}$, where (x, y) is the coordinate of the tip.

We derive the probabilistic safe set in Fig. 5.4 under the assuming no friction. This is not the case in a Mujoco-based simulation, but the effect of such

an assumption is minor. Recall that the states displayed in Fig. 5.4 stand for an end effector with zero velocity. If appropriate control is applied, the robot can avoid reaching the target set by moving toward an arbitrary position near the goal unless it launched from the target set at the beginning.

In our simulation studies, we only assess the agents with the states where the goal point is given by $(-2l, 0)$, and the angular velocity is $(\dot{\theta}_1, \dot{\theta}_2) = (0, 0)$. We use the Reacher configuration provided by Gym [34].

Environment	Time limit	γ
Integrator	1000	$1 - 10^{-4}$
Reacher	300	$1 - 10^{-3}$

Table 5.2 The environment-specific parameters.

Chapter 6

Conclusion

We have proposed a model-free safety specification tool that incorporates a Lyapunov-based safe RL approach with probabilistic reachability analysis. Our method exploits the Lyapunov constraint to construct an exploratory policy that mitigates the discrepancy between state distributions of the experience replay (or the tabular Q-function) and the environment. Another salient feature of the proposed method is that it can be implemented on generic, model-free deep RL algorithms, particularly in continuous state and action spaces through Lagrangian relaxation. The results of our experiments demonstrate that our method encourages visiting the unspecified states, thereby improving the accuracy of specification. By bridging probabilistic reachability analysis and reinforcement learning, this work can provide an exciting avenue for future research in terms of extensions to partially observable MDPs, and model-based exploration and its regret analysis, among others.

Bibliography

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, Nov 2008.

[2] A. Majumdar, R. Vasudevan, M. M. Tobenkin, and R. Tedrake, "Convex optimization of nonlinear feedback controllers via occupation measures," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1209–1230, 2014.

[3] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, "A general system decomposition method for computing reachable sets and tubes," *IEEE Trans. Automat. Contr.*, vol. 63, no. 11, pp. 3675–3688, 2018.

[4] J. H. Gillula, G. M. Hoffmann, H. Huang, M. P. Vitus, and C. J. Tomlin, "Applications of hybrid reachability analysis to robotic aerial vehicles," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 335–354, 2011.

[5] G. Piovan and K. Byl, "Reachability-based control for the active slip model," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 270–287, 2015.

[6] N. Malone, H. T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1124–1138, 2017.

[7] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Trans. Automat. Contr.*, vol. 64, no. 7, pp. 2737–2752, 2018.

[8] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. Tomlin, "Bridging Hamilton-Jacobi safety analysis and reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 8550–8556.

[9] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Adv. Neural Inf. Process. Syst.*, 2017.

[10] S. M. Richards, F. Berkenkamp, and A. Krause, "The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proc. 2nd Conf. on Robot Learn.*, 2018, pp. 466–476.

[11] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2460–2465.

[12] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," *arXiv preprint arXiv:1912.10099*, 2019.

[13] S. Summers and J. Lygeros, "Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem," *Automatica*, vol. 46, no. 12, pp. 1951–1961, 2010.

[14] K. Lesser and M. Oishi, "Approximate safety verification and control of partially observable stochastic hybrid systems," *IEEE Trans. Automat. Contr.*, vol. 62, no. 1, pp. 81–96, 2017.

[15] I. Yang, "A dynamic game approach to distributionally robust safety specifications for stochastic systems," *Automatica*, vol. 94, pp. 94–101, 2018.

[16] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Adv. Neural Inf. Process. Syst.*, 2018, pp. 8103–8112.

[17] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," in *Adv. Neural Inf. Process. Syst.*, 2016, pp. 4312–4320.

[18] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. AAAI Conf. on Artif. Intell.*, 2018, pp. 2669–2678.

[19] A. Wachi, Y. Sui, Y. Yue, and M. Ono, "Safe exploration and optimization of constrained MDPs using Gaussian processes," in *Proc. AAAI Conf. on Artif. Intell.*, 2018, pp. 6548–6556.

[20] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," in *Adv. Neural Inf. Process. Syst.* Curran Associates, Inc., 2019, pp. 3240–3249.

[21] R. Cheng, G. Orosz, R. Murray, and J. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous

control tasks," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, vol. 33. AAAI Press, 2019, pp. 3387–3395.

[22] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *J. Artif. Int. Res.*, vol. 24, no. 1, p. 81–108, Jul 2005.

[23] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Int. Conf. on Mach. Learn.*, vol. 70. PMLR, 2017, pp. 22–31.

[24] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *arXiv preprint arXiv:1805.11074*, 2018. [Online]. Available: http://arxiv.org/abs/1805.11074

[25] M. Wen and U. Topcu, "Constrained cross-entropy method for safe reinforcement learning," in *Adv. Neural Inf. Process. Syst.* Curran Associates, Inc., 2018, pp. 7450–7460.

[26] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Mach. Learn.*, vol. 16, no. 3, pp. 185–202, 1994.

[27] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA, USA: Athena Scientific, 1999.

[28] S. Bohez, A. Abdolmaleki, M. Neunert, J. Buchli, N. Heess, and R. Hadsell, "Value constrained model-free continuous control," *arXiv preprint arXiv:1902.04623*, 2019.

[29] H. van Hasselt, "Double Q-learning," in *Adv. Neural Inf. Process. Syst.*, 2010, pp. 2613–2621.

[30] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Int. Conf. on Mach. Learn.*, vol. 48, 2016, pp. 1329–1338.

[31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *arXiv preprint arXiv:1509.02971*, 2015.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.

[33] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[34] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

초록

자율주행, 로봇 수술 등 자율시스템 및 로보틱스의 떠오르는 응용 분야의 절대 다수는 안전한 동작을 보장하기 위해 일정한 제약을 필요로 한다. 특히, 안전제약은 시스템 모델에 대해 제한된 정보만 알려져 있을 때에도 보장되어야 한다. 이에 따라, 본 논문에서는 확률적 도달성 분석(probabilistic reachability analysis)과 안전 강화학습(safe reinforcement learning)을 결합하여 시스템이 안전하게 동작할 확률의 최댓값으로 정의되는 안전 사양을 별도의 모델 없이 추정하는 방법론을 제안한다. 우리의 접근법은 매번 정책을 새로 구하는 과정에서 그 결과물이 안전함에 대한 기준을 충족시키도록 제한을 거는 것으로, 이를 위해 안전한 정책에 관한 랴푸노프 함수를 구축한다. 그 결과로 산출되는 일련의 정책으로부터 안전 집합(safe set)이라 불리는 안전한 동작이 보장되는 영역이 계산되고, 이 집합은 단조롭게 확장하여 점차 최적해로 수렴하도록 만다. 또한, 우리는 조사되지 않은 상태의 안전성을 더 빠르게 파악할 수 있는 효율적인 안전 탐사 체계를 개발하였다. 랴푸노브 차폐를 이용한 결과, 우리가 제안하는 탐험 정책은 높은 확률로 위험하다 여겨지는 상태를 피하도록 제한이 걸린다. 여기에 더해 우리는 고차원 시스템을 처리하기 위해 제안한 방법을 심층강화학습으로 확장했고, 구현 가능한 액터-크리틱 알고리즘을 만들기 위해 라그랑주 이완법을 사용하였다. 더불어 본 방법의 실효성은 연속적인 제어 벤치마크인 2차원 평면에서 동작하는 2-DOF 로봇 팔을 통해 실험적으로 입증되었다.

주요어: 안전 강화학습, 확률적 도달성 분석, 안전성 상세화
학번: 2018-25960

Acknowledgements

First and foremost, I would like to show gratitude to my supervisor, Insoon Yang. This work would have never been accomplished without his involvement during the past two years. For every stage of the research process, he provided constructive feedbacks when I needed them most. His support always helped me developing the idea, improving the experiment results, and revising the articles.

I wish to express my sincere thanks to my colleagues, Taewoo Kang, Astghik Hakobyan, Sihyun Choi and Melike Ermis for their assistance throughout the course of this thesis. Not only have they spared the time for discussion, but also they have participated in solving technical issues without hesitation. I am also grateful to Kuikam Kwon in the Department of Mechanical Engineering for offering me the valuable co-working experience. Special thanks to a friend of mine, Hyunjun Kim, for both the personal and professional support whenever I confronted hardships.

Last but not least, I should express gratitude to my parents and my brother for the endless dedication and continuous encouragement. This work stands as a tribute for their kindness and love.